Lab #2 Answers

2021-02-08

Contents

Part 1 Exercises

Exercises with CO_2 Data from the Mauna Loa Observatory

Using the `select` function, make a new data tibble called `mlo_seas`, from the original `mlo_data`, which only has two columns: `date` and `co2.seas`, where `co2.seas` is a renamed version of `co2.filled.seas` from the original tibble.

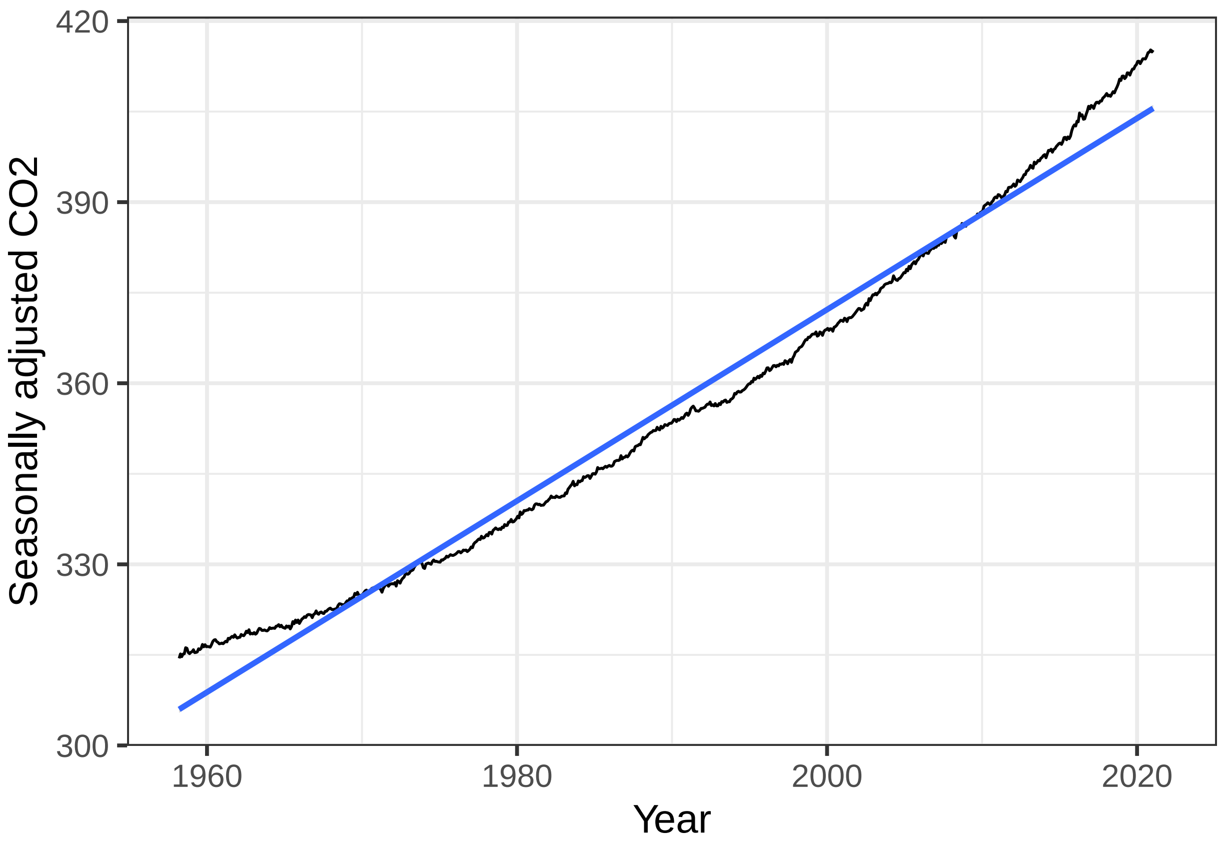
```
mlo_seas = select(mlo_data, date, co2.seas = co2.filled.seas)
```

Now plot this with `co2.seas` on the *y* axis and `date` on the *x* axis, and a linear fit:

```
ggplot(mlo_seas, aes(x = date, y = co2.seas)) +
  geom_line() +
  geom_smooth(method = "lm") +
  labs(x = "Year", y = "Seasonally adjusted CO2")
```

```
## Warning: Removed 13 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 13 row(s) containing missing values (geom_path).
```



Now fit a linear function to find the annual trend of `co2.seas`. Save the results of your fit in a variable called `trend.seas`.

```
trend.seas = lm(co2.seas ~ date, data = mlo_seas)
seasonal_trend = coef(trend.seas)['date']
```

Compare the trend you fit to the raw `co2.filled` data to the trend you fit to the seasonally adjusted data.

Answer: The trend in the seasonally adjusted data is 1.58 ppm CO2 per year, which is very close to the trend we fit to the raw data (rformat_md(co2_trend, digits = 2)‘).

Exercises with Global Temperature Data from NASA

We can also download a data set from NASA's Goddard Institute for Space Studies (GISS), which contains the average global temperature from 1880 through the present.

The URL for the data file is https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv

Download this file and save it in the directory `_data/global_temp_land_sea.csv`.

```
giss_temp_file = "_data/global_temp_land_sea.csv"
download.file(giss_url, giss_temp_file)
```

- Open the file in Excel or a text editor and look at it.
- Unlike the CO_2 data file, this one has a single line with the data column names, so you can specify `col_names=TRUE` in `read_csv` instead of having to write the column names manually.

- How many lines do you have to tell `read_csv` to skip?
- `read_csv` can automatically figure out the data types for each column, so you don't have to specify `col_types` when you call `read_csv`
- This file uses `***` to indicate missing values instead of `-99.99`, so you will need to specify `na="***"` in `read_csv`.

  For future reference, if you have a file that uses multiple different values to indicate missing values, you can give a vector of values to `na` in `read_csv`: `na = c('***','-99.99', 'NA', '')` would tell `read_csv` that if it finds any of the values "***“,”-99.99“,”NA", or just a blank with nothing in it, any of those would correspond to a missing value, and should be indicated by `NA` in R.

Now read the file into R, using the `read_csv` function, and assign the resulting tibble to a variable `giss_temp`

```r
giss_temp = read_csv(giss_temp_file, skip = 1,
                     na = c('***', '-99.99', 'NA', ''))
head(giss_temp, 5)
```

```
## # A tibble: 5 x 19
##    Year   Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec ‘J-D‘ ‘D-N‘   DJF
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dl
## 1  1880 -0.17 -0.23 -0.08 -0.15 -0.08 -0.19 -0.17 -0.08 -0.13 -0.22 -0.2  -0.16 -0.15 NA    NA    -0
## 2  1881 -0.18 -0.13  0.04  0.06  0.08 -0.17  0.02 -0.02 -0.14 -0.2  -0.17 -0.05 -0.07 -0.08 -0.16  0
## 3  1882  0.18  0.15  0.06 -0.15 -0.13 -0.21 -0.15 -0.06 -0.13 -0.22 -0.15 -0.34 -0.1  -0.07  0.09 -0
## 4  1883 -0.28 -0.35 -0.11 -0.17 -0.16 -0.06 -0.05 -0.12 -0.2  -0.1  -0.21 -0.1  -0.16 -0.18 -0.32 -0
## 5  1884 -0.11 -0.06 -0.35 -0.39 -0.32 -0.35 -0.31 -0.26 -0.26 -0.24 -0.32 -0.3  -0.27 -0.25 -0.09 -0
```

Something is funny here: Each row corresponds to a year, but there are columns for each month, and some extra columns called “J-D”, “D-N”, “DJF”, “MAM”, “JJA”, and “SON”. These stand for average values for the year from January through December, the year from the previous December through November, and the seasonal averages for Winter (December, January, and February), Spring (March, April, and May), Summer (June, July, and August), and Fall (September, October, and November).

The temperatures are recorded not as the thermometer reading, but as *anomalies*. If we want to compare how temperatures are changing in different seasons and at different parts of the world, raw temperature measurements are hard to work with because summer is hotter than winter and Texas is hotter than Alaska, so it becomes difficult to compare temperatures in August to temperatures in January, or temperatures in Texas to temperatures in Alaska and tell whether there was warming.

To make it easier and more reliable to compare temperatures at different times and places, we define anomalies: The temperature anomaly is the difference between the temperature recorded at a certain location during a certain month and a baseline reference value, which is the average temperature for that month and location over a period that is typically 30 years.

The GISS temperature data uses a baseline reference period of 1951–1980, so for instance, the temperature anomaly for Nashville in July 2017 would be the monthly average temperature measured in Nashville during July 2017 minus the average of all July temperatures measured in Nashville from 1951–1980.

The GISS temperature data file then averages the temperature anomalies over all the temperature-measuring stations around the world and reports a global average anomaly for every month from January 1880 through the latest measurements available (currently, July 2017).

Let's focus on the months only. Use `select` to select just the columns for “Year” and January through December (if you are selecting a consecutive range of columns between “Foo” and “Bar” in the tibble `df`, you can call `select(df, Foo:Bar)` or `df %>% select(Foo:Bar)`). Save the result in a variable called `giss_monthly`

```r
giss_monthly = select(giss_temp, Year:Dec)
```

Next, it will be difficult to plot all of the data if the months are organized as columns. What we want is to transform the data tibble into one with three columns: "year", "month", and "anomaly". We can do this easily using the `pivot_longer` function from the tidyverse package: `pivot_longer(df, cols = -Year, names_to = "month", values_to = "anomaly")` or `df %>% pivot_longer(cols = -Year, names_to = "month", values_to = "anomaly")` will gather all of the columns except `Year` (the minus sign in the argument to `cols` means to include all columns except the ones indicated with a minus sign) and:

- Make a new tibble with three columns: "Year", "month", and "anomaly"
- For each row in the original tibble, make rows in the new tibble for each of the columns "Jan" through "Dec", putting the name of the column in "month" and the anomaly in "anomaly".

Here is an example of using `pivot_longer`, using a data set of quarterly approval ratings for U.S. presidents from 1945–1974:

```r
df = read_rds(file.path(data_dir, "president-approval.Rds"))

print("First 10 rows of df are")
```

```
## [1] "First 10 rows of df are"
```

```r
print(head(df, 10))
```

```
## # A tibble: 10 x 5
##     year    Q1    Q2    Q3    Q4
##    <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  1945    NA    87    82    75
##  2  1946    63    50    43    32
##  3  1947    35    60    54    55
##  4  1948    36    39    NA    NA
##  5  1949    69    57    57    51
##  6  1950    45    37    46    39
##  7  1951    36    24    32    23
##  8  1952    25    32    NA    32
##  9  1953    59    74    75    60
## 10  1954    71    61    71    57
```

For each year, the table has a column for the year and four columns (Q1 … Q4) that hold the quarterly approval ratings for the president in that quarter. Now we want to organize these data into three columns: one column for the year, one column to indicate the quarter, and one column to indicate the approval rating.

We do this with the `pivot_longer` function.

```r
dfg = df %>%
  pivot_longer(cols = -year, names_to = "quarter", values_to = "approval") %>%
  arrange(year, quarter)

# the pivot_longer command organizes the data into tidy columns:
# names_to = "quarter" tells pivot_longer to create a column called "quarter"
# and store the names of the original columns there.
# values_to = "approval" tells pivot_longer to create a column called "approval"
# and store the values from the columns there.
# cols = -year tells pivot_longer NOT to change the column "year".
# So the approval ratings from the second quarter of 1960 will be stored in
# a row where the column year contains 1960, quarter contains "Q2", and
# approval contains the approval rating.
#
# the arrange command sorts the rows of the resulting tibble to put
# the years in ascending order, from 1945 to 1971, and within each year,
# sort the quarters in alphabetical order from Q1 to Q4

head(dfg) # print the first few rows of the tibble.
```

```
## # A tibble: 6 x 3
##    year quarter approval
##   <dbl> <chr>      <dbl>
## 1  1945 Q1            NA
## 2  1945 Q2            87
## 3  1945 Q3            82
## 4  1945 Q4            75
## 5  1946 Q1            63
## 6  1946 Q2            50
```

Now you try to do the same thing to the `giss_monthly` data. Use `pivot_longer` to re-organize the data to have three columns: one column for the year, one column called “month” for the name of the month, and one column called “anomaly” for the temperature anomaly in that month. Store the result in a new variable called `giss_g`

```r
giss_g = giss_monthly %>%
  pivot_longer(cols = -Year, names_to = "month", values_to = "anomaly")
```

Remember how the CO_2 data had a column `date` that had a year plus a fraction that corresponded to the month, so June 1960 was 1960.4548?

Here is a trick that lets us do the same for the `giss_g` data set. R has a data type called `factor` that it uses for managing categorical data, such as male versus female, Democrat versus Republican, and so on. Categorical factors have a textual label, but behind the scenes, R thinks of them as integer numbers. Normal factors don’t have a special order, so R sorts the values alphabetically. However, there is another kind of factor called an ordered factor, which allows us to specify the order of the values.

We can use a built-in R variable called `month.abb`, which is a vector of abbreviations for months.

The following command will convert the `month` column in `giss_g` into an ordered factor that uses the integer values 1, 2, . . . , 12 to stand for “Jan”, “Feb”, . . . , “Dec”, and then uses those integer values to create a new column, `date` that holds the fractional year, just as the `date` column in `mlo_data` did:

```r
giss_g = giss_g %>%
  mutate(month = ordered(month, levels = month.abb),
         date = Year + (as.integer(month) - 0.5) / 12) %>%
  arrange(date)`
```

In the code above, `ordered(month, levels = month.abb)` converts the variable `month` from a character (text) variable that contains the name of the month to an ordered factor that associates a number with each month name, such that ‘Jan’ = 1 and ‘Dec’ = 12.

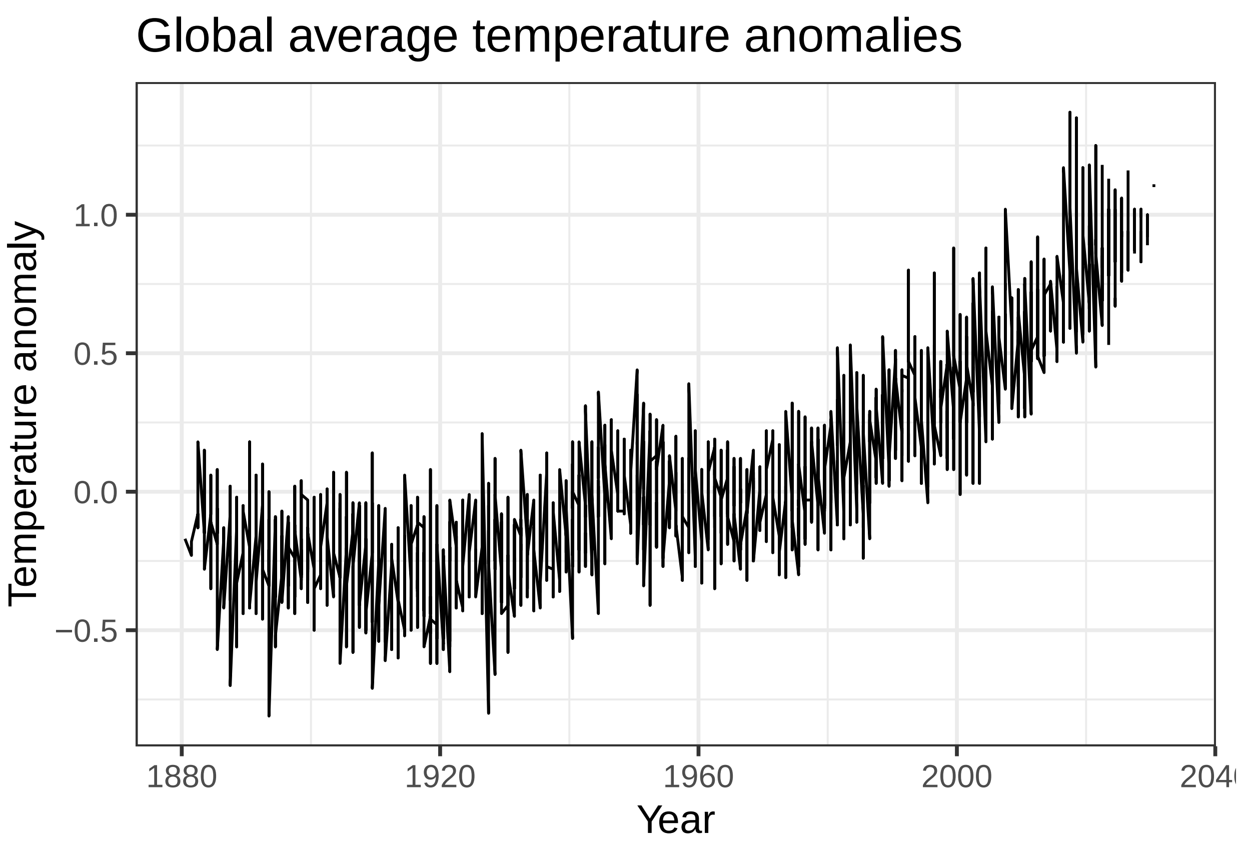
Then we create a new column called `date` to get the fractional year corresponding to that month. We have to explicitly convert the ordered factor into a number using the function `as.integer()`, and we subtract 0.5 because the time that corresponds to the average temperature for the month is the middle of the month.

Below, use code similar to what I put above to add a new `date` column to `giss_g`.

```
giss_g = giss_g %>%
  mutate(month = ordered(month, levels = month.abb),
         date = Year + (as.integer(month) - 0.5)) %>%
  arrange(date)
```

Now plot the monthly temperature anomalies versus date:

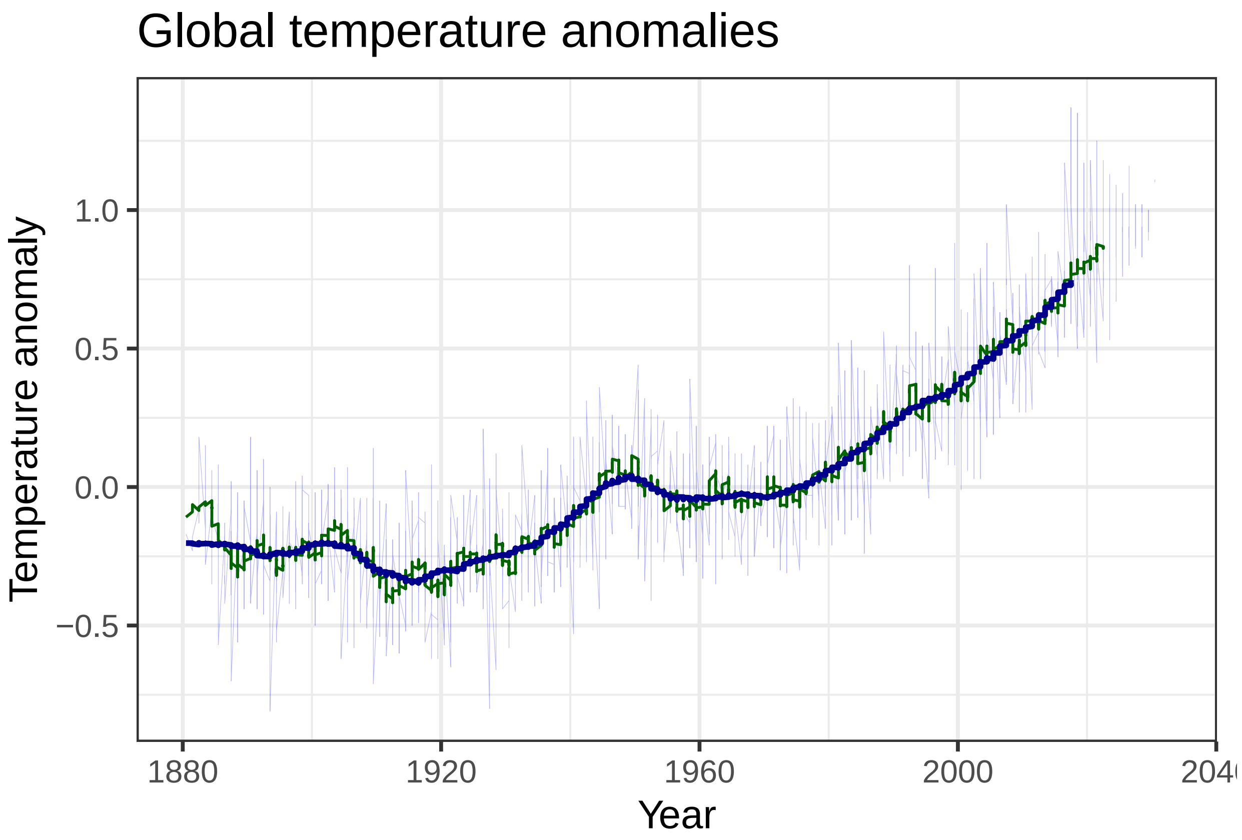
```
ggplot(giss_g, aes(x = date, y = anomaly)) +
  geom_line() +
  labs(x = "Year", y = "Temperature anomaly",
       title = "Global average temperature anomalies")
```



That plot probably doesn't look like much, because it's very noisy. Use the function `slide_vec` from the package `slider` to create new columns in `giss_g` with 12-month and 10-year (i.e., 120-month) sliding averages of the anomalies.

Make a new plot in which you plot a thin blue line for the monthly anomaly (use `geom_line(aes(y = anomaly), color = "blue", alpha = 0.3, size = 0.1)`; alpha is an optional specification for transparency where 0 means invisible (completely transparent) and 1 means opaque), a medium dark green line for the one-year sliding average, and a thick dark blue line for the ten-year sliding average.

```
giss_g %>%
  mutate(annual = slide_vec(anomaly, mean, .before = 5, .after = 6),
         decadal = slide_vec(anomaly, mean, .before = 59, .after = 60)) %>%
  ggplot(aes(x = date)) +
  geom_line(aes(y = anomaly), color = "blue", alpha = 0.3, size = 0.1) +
  geom_line(aes(y = annual), color = "darkgreen", size = 0.5) +
  geom_line(aes(y = decadal), color = "darkblue", size = 1) +
  labs(x = "Year", y = "Temperature anomaly",
       title = "Global temperature anomalies")
```



The graph shows that temperature didn't show a steady trend until starting around 1970, so we want to isolate the data starting in 1970 and fit a linear trend to it.

To select only rows of a tibble that match a condition, we use the function `filter` from the `tidyverse` package:

`data_subset = df %>% filter( conditions )`, where `df` is your original tibble and `conditions` stands for whatever conditions you want to apply. You can make a simple condition using equalities or inequalities:

- `data_subset = df %>% filter( month == "Jan")` to select all rows where the month is "Jan"
- `data_subset = df %>% filter( month != "Aug")` to select all rows where the month is not August.
- `data_subset = df %>% filter( month %in% c("Sep", "Oct", "Nov")` to select all rows where the month is one of "Sep", "Oct", or "Nov".
- `data_subset = df %>% filter(year >= 1945)` to select all rows where the year is greater than or equal to 1945.
- `data_subset = df %>% filter(year >= 1951 & year <= 1980 )` to select all rows where the year is between 1951 and 1980, inclusive.
- `data_subset = df %>% filter(year >= 1951 | month == "Mar")` to select all rows where the year is greater than or equal to 1951 or the month is "Mar". this will give all rows from January 1951 onward, plus all rows before 1951 where the month is March.

Below, create a new variable `giss_recent` and assign it a subset of `giss_g` that has all the data from January 1970 through the present. Use the `lm` function to calculate the linear trend for the monthly anomaly and report it. Remember that if you use `lm` to calculate a linear trend, you can get the value for the slope using the `coef` function:

Example:

```
co2_fit = lm(co2 ~ date, data = mlo_simple)
co2_slope = coef(co2_fit)['data']
```

What is the average change in temperature from one year to the next?

```
giss_recent = giss_g %>% filter(Year >= 1970)

temp_fit = lm(anomaly ~ date, data = giss_recent)

temp_trend = coef(temp_fit)["date"]
```

Answer: On average, the temperature rose by 0.018 degree Celsius per year.

Did Global Warming Stop after 1998?

It is a common skeptic talking point that global warming stopped in 1998. In years with strong El Niños, global temperatures tend to be higher and in years with strong La Niñas, global temperatures tend to be lower. We will discuss why later in the semester.

The year 1998 had a particularly strong El Niño, and the year set a record for global temperature that was not exceeded for several years. Indeed, compared to 1998, it might look as though global warming paused for many years.

We will examine whether this apparent pause has scientific validity.

To begin with, we will take the monthly GISS temperature data and convert it to annual average temperatures, so we can deal with discrete years, rather than separate temperatures for each month.

We do this with the `group_by` and `summarize` functions.

We also want to select only recent data, so we arbitrarily say we will look at temperatures starting in 1979, which gives us 19 years before the 1998 El Ni~o.

If we go back to the original `giss_g` data tibble, run the following code:

```
giss_annual = giss_g %>%
  filter(Year >= 1979) %>%
  group_by(Year) %>%
  summarize(anomaly = mean(anomaly)) %>%
  ungroup() %>%
  mutate(date = Year + 0.5, before = Year < 1998)
```

This code groups the giss data by the year, so that one group will have January–December 1979, another will have January–December 1980, and so forth.

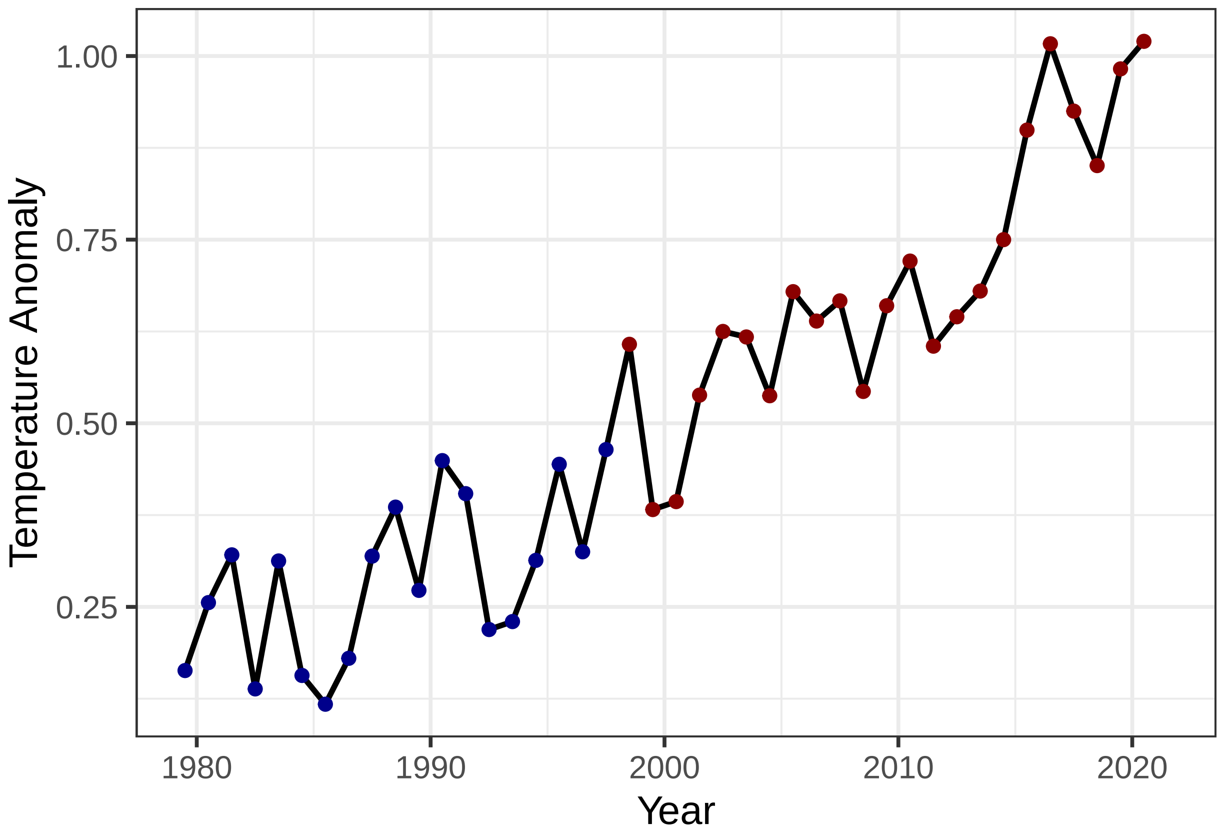
Then we replace the groups of 12 rows for each year (each row represents one month) with a single row that represents the average of those 12 months.

It is important to tell R to `ungroup` the data after we're done working with the groups.

Finally, we set `date` to `year + 0.5` because the average of a year corresponds to the middle of the year, not the beginning and we introduce a new column `before`, which indicates whether the data is before the 1998 El Niño:

Now plot the data and color the points for 1998 and afterward dark red to help us compare before and after 1998.

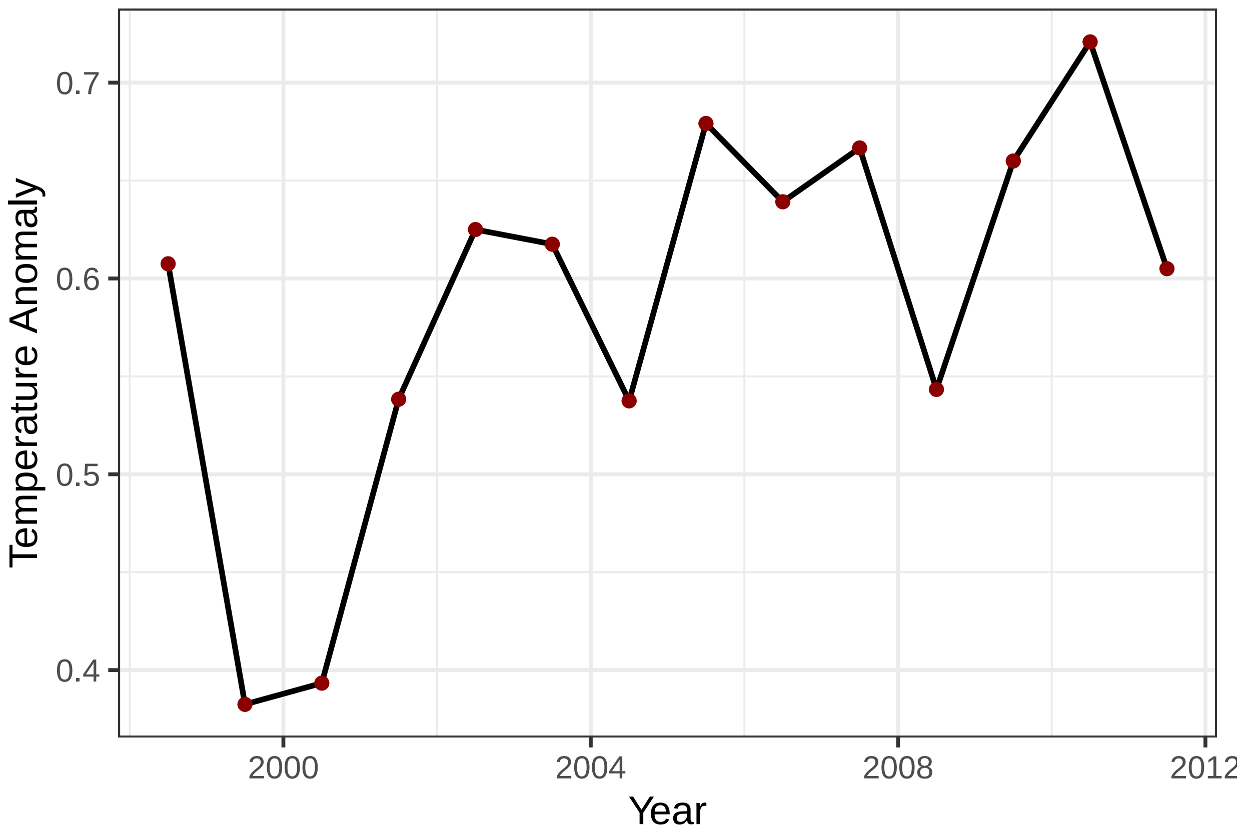
```
ggplot(giss_annual, aes(x = date, y = anomaly)) +
  geom_line(size = 1) +
  geom_point(aes(color = before), size = 2) +
  scale_color_manual(values = c("TRUE" = "darkblue", "FALSE" = "darkred"),
                     guide = "none") +
  # ^^^ color "before" points dark blue, "after" points dark red.
  # guide = "none" tells ggplot not to show a legend explaining the colors.
  labs(x = "Year", y = "Temperature Anomaly")
```



Does it look as though the red points are not rising as fast as the blue points?

Let's just plot the data from the years 1998–2011. Use the `filter` function to select just the date from the years 1998–2011 and pass that to `ggplot`.

```
giss_annual %>%
  filter(Year >= 1998, Year <= 2011) %>%
  ggplot(aes(x = date, y = anomaly)) +
  geom_line(size = 1) +
  geom_point(aes(color = before), size = 2) +
  scale_color_manual(values = c("TRUE" = "darkblue", "FALSE" = "darkred"),
                     guide = "none") +
  # ^^^ color "before" points dark blue, "after" points dark red.
  # guide = "none" tells ggplot not to show a legend explaining the colors.
  labs(x = "Year", y = "Temperature Anomaly")
```



Now how does it look?

Let's use the `filter` function to break the data into two different data sets, which we will store in tibbles called `giss_before` and `giss_after`: `giss_before` will have the data from 1979–1998 and the other, `giss_after` will have the data from 1998 onward (note that the year 1998 appears in both data sets).

Also, use the `mutate` function to add a column called `timing` to each of the split data sets and set the value of this column to "Before" for `giss_before` and "After" for `giss_after`.

```
giss_before = giss_annual %>% filter(Year <= 1998) %>%
  mutate(timing = "Before")
giss_after = giss_annual %>% filter(Year >= 1998) %>%
  mutate(timing = "After")
```

Now use `lm` to find the trend in temperature data in `giss_before` (from 1979–1998) and assign it to a variable `giss_trend`.

```
fit_before = lm(anomaly ~ date, data = giss_before)
trend_before = coef(fit_before)['date']
```

Next, combine the two tibbles into one tibble, using the `bind_rows` function. If you have created the `giss_before` and `giss_after` tibbles, then you can un-comment the code below to combine them.

```
giss_combined = bind_rows(giss_before, giss_after)
```

Now let's use ggplot to plot `giss_combined`:

- Aesthetic mapping:

  - Use the `date` column for the *x* variable.
  - Use the `anomaly` column for the *y* variable.
  - Use the `timing` column to set the color of plot elements

- Plot both lines and points.

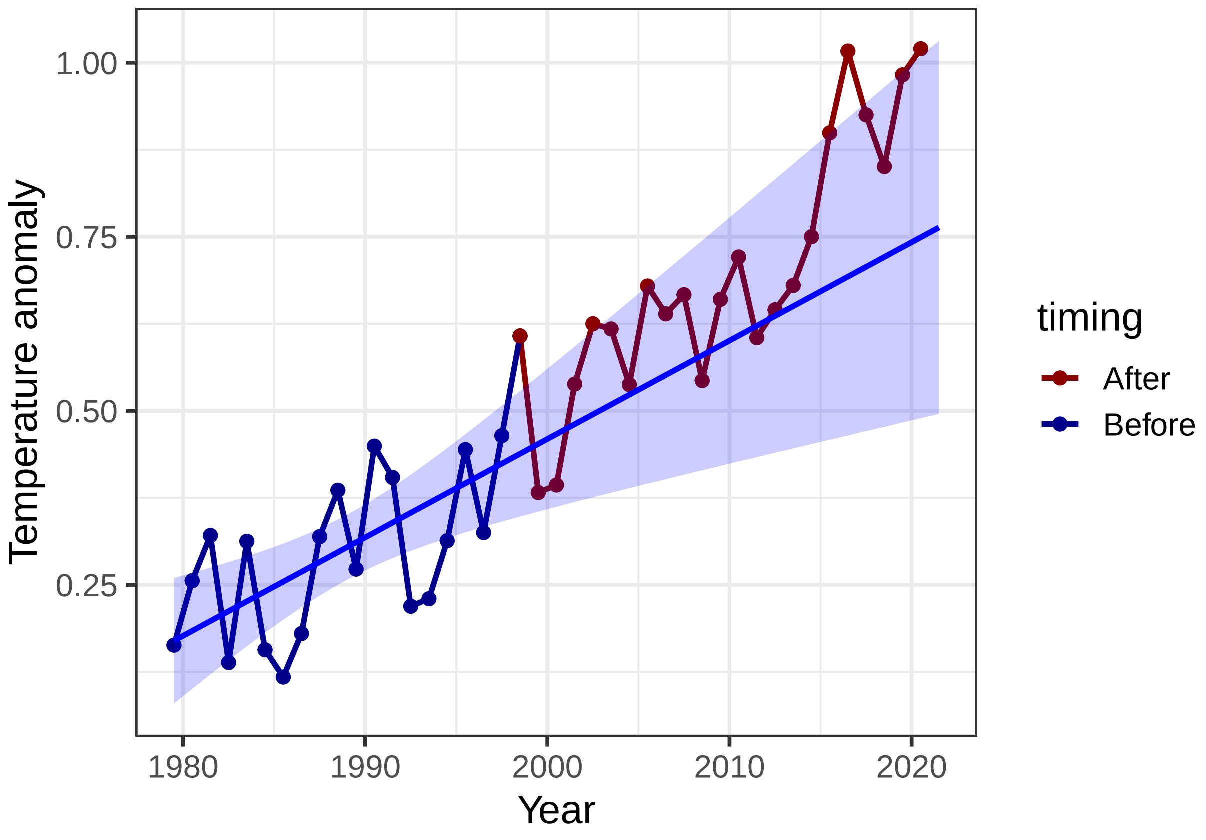
  - Set the `size` of the lines to 1
  - Set the `size` of the points to 2

- Use the `scale_color_manual` function to set the color of "Before" to "darkblue" and "After" to "darkred"
- Use `geom_smooth(data = giss_before, method="lm", color = "blue", fill = "blue", alpha = 0.2, fullrange = TRUE)` to show a linear trend that is fit just to the `giss_before` data.

```
ggplot(giss_combined, aes(x = date, y = anomaly, color = timing)) +
  geom_line(size = 1) +
  geom_point(size = 2) +
  geom_smooth(data = giss_before, method = "lm", color = "blue",
              fill = "blue", alpha = 0.2, fullrange = TRUE) +
  scale_color_manual(values = c("Before" = "darkblue", "After" = "darkred")) +
  labs(x = "Year", y = "Temperature anomaly")
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Try this with the parameter `fullrange` set to `TRUE` and `FALSE` in the `geom_smooth` function. What is the difference?

What this plot shows is the full data set, and a linear trend that is fit just to the "before" data. The trend line shows both the best fit for a trend (that's the solid line) and the range of uncertainty in the fit (that's the light blue shaded area around the line).

If the temperature trend changed after 1998 (e.g., if the warming paused, or if it reversed and started cooling) then we would expect the temperature measurements after 1998 to fall predominantly below the extrapolated trend line, and our confidence that the trend had changed would depend on the number of points that fall below the shaded uncertainty range.

How many of the red points fall below the trend line?

Answer: Out of 23 red points (the years 1998–2020), four fall below the trend line

How many of the red points fall above the trend line?

Answer: 19 of the 23 red points fall above the trend line.

If we just look at the years 1998–2012, how many of the red points fall above vs. below the trend line?

Answer: From 1998–2012, 4 points fall below the trend line and 11 fall above it.

What do you conclude about whether global warming paused or stopped for several years after 1998?

Answer: Even if we just look at the years 1998–2012, it is clear that most years were warmer than we would have expected if the warming had followed its historical trend from 1979–1998. This means that if anything, the earth was warming faster than before—the opposite of pausing or stopping. After 2012, all of the points are above the trend line. Most of them are far above it. So there is no reasonable way to look at this data and conclude that it stopped warming, or even paused temporarily, after 1998.

Part 2 Exercises from Chapter 3

- **All students** do Chapter 3, exercises 2–3.
- **Graduate students** should also do Chapter 3, exercise 1.

For the exercises, use the following numbers:

- $I_{solar} = 1350$ W/m^2
- $\sigma = 5.67 \times 10^{-8}$
- $\alpha = 0.30$
- $\varepsilon = 1.0$

```
I_solar = 1350
alpha = 0.30
epsilon = 1.0
sigma = 5.67E-8
```

Exercise 3.1 (Grad. students only)

The moon with no heat transport. The Layer Model assumes that the temperature of the body in space is all the same. This is not really very accurate, as you know that it is colder at the poles than it is at the equator. For a bare rock with no atmosphere or ocean, like the moon, the situation is even worse because fluids like air and water are how heat is carried around on the planet. So let us make the other extreme assumption, that there is no heat transport on a bare rock like the moon. Assume for comparability that the albedo of this world is 0.30, same as Earth's.

What is the equilibrium temperature of the surface of the moon, on the equator, at local noon, when the sun is directly overhead? (Hint: First figure out the intensity of the local solar radiation I_{solar})

```
albedo_moon = 0.3
epsilon_moon = 1
I_moon = I_solar * (1 - albedo_moon)
T_moon = (I_moon / (epsilon_moon * sigma))^0.25
```

Answer: The temperature of the equator of the moon at noon is 359.3 K.

> What is the equilibrium temperature on the dark side of the moon?

Answer: The intensity of sunlight on the dark side of the moon is zero, so the temperature would be zero Kelvin.

Exercise 3.2

> **A Two-Layer Model.** Insert another atmospheric layer into the model, just like the first one. The layer is transparent to visible light but a blackbody for infrared.

```
make_layer_diagram(2)
```

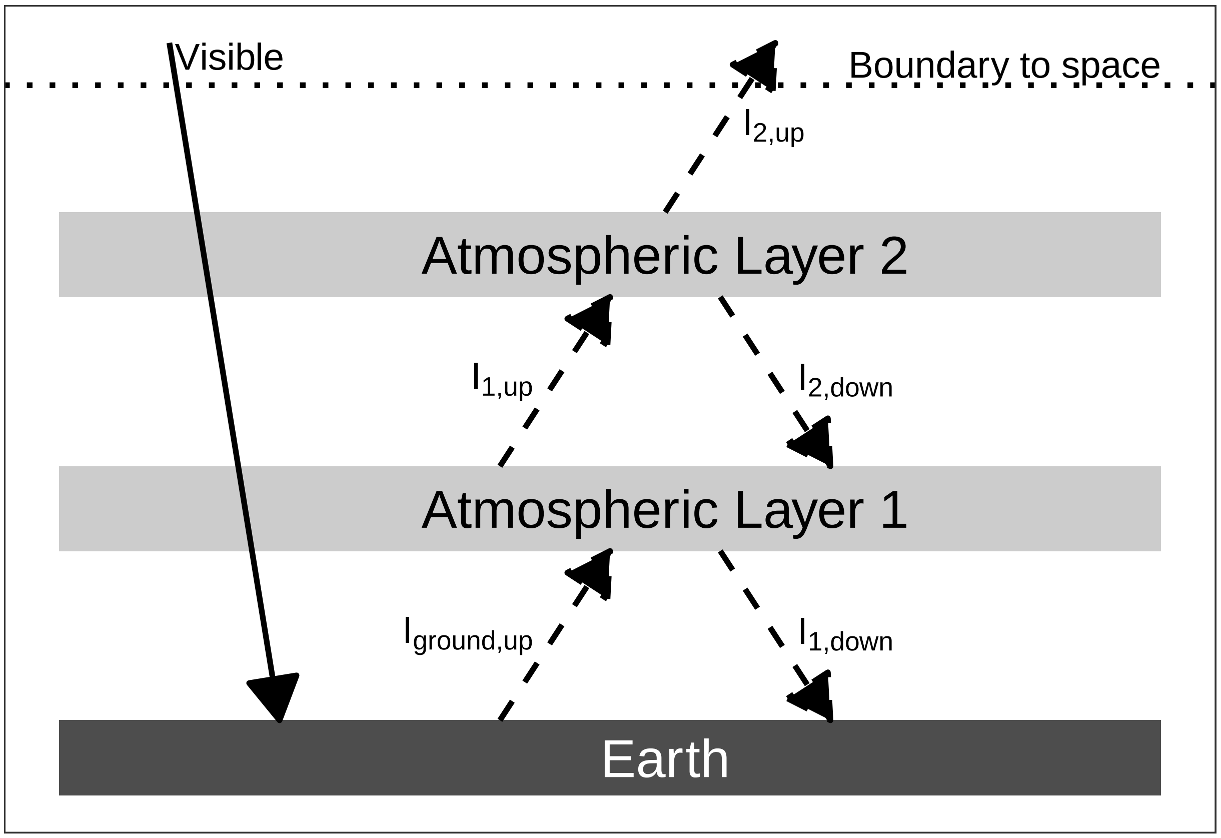


Figure 1: An energy diagram for a planet with two panes of glass for an atmosphere. The intensity of absorbed visible light is $(1-\alpha)I_{solar}/4$.

a) Write the energy budgets for both atmospheric layers, for the ground, and for the Earth as a whole, like we did for the One-Layer Model.

Answer:

- Top of the atmosphere: $I_{2,\text{up}} = I_{\text{visible}} = (1-\alpha)I_{\text{solar}}/4$
- Layer 2: $I_{1,\text{up}} = I_{2,\text{up}} + I_{2,\text{down}}$
- Layer 1: $I_{\text{ground,up}} + I_{2,\text{down}} = I_{1,\text{up}} + I_{1,\text{down}}$
- Ground: $I_{\text{ground,up}} = I_{\text{visible}} + I_{1,\text{down}}$

b) Manipulate the budget for the Earth as a whole to obtain the temperature T_2 of the top atmospheric layer, labeled Atmospheric Layer 2 in the figure above. Does this part of the exercise seem familiar in any way? Does the term ring any bells?

Top of the atmosphere:

$$
\begin{aligned}
I_{2,\text{up}} &= I_{\text{visible}} = (1-\alpha)I_{\text{solar}}/4 \\
I_{2,\text{up}} &= \varepsilon\sigma T_2^4 \\
T_2 &= \sqrt[4]{\frac{(I_{2,\text{up}}}{\varepsilon\sigma}} \\
&= \sqrt[4]{\frac{(1-\alpha)I_{\text{solar}}}{4\varepsilon\sigma}}
\end{aligned}
$$

This is the same as the bare-rock temperature.

```
epsilon = 1.0
alpha = 0.30
I_solar = 1350
I_visible = (1 - alpha) * I_solar / 4
I_2_up = I_visible
T_2 = (I_2_up / (epsilon * sigma))^0.25
```

Answer: The temperature of layer 2 is 254.1 K, which is the same as the bare-rock temperature. In layer models, the top layer of the atmosphere is *always* the bare-rock temperature.

c) Insert the value you found for T_2 into the energy budget for layer 2, and solve for the temperature of layer 1 in terms of layer 2. How much bigger is T_1 than T_2?

From the energy budget for Layer 2, $I_{1,\text{up}}$ = $I_{2,\text{up}}$ + $I_{2,\text{down}}$. The temperature of the bottom of the layer is the same as the temperature for the top of the layer, so $I_{2,\text{down}}$ = $I_{2,\text{up}}$

```
I_1_up = 2 * I_2_up
T_1 = (I_1_up / (epsilon * sigma))^0.25
```

You could also let R do more of the work for you by writing

```
I_2_down = I_2_up
I_1_up = I_2_up + I_2_down
T_1 = (I_1_up / (epsilon * sigma))^0.25
```

Answer: The temperature of layer 1 is 302.1 K. This is the same as the ground temperature in a 1-layer model.

d) Now insert the value you found for T_1 into the budget for atmospheric layer 1 to obtain the temperature of the ground, T_{ground}. Is the greenhouse effect stronger or weaker because of the second layer?

From the energy budget for layer 1,

- $I_{\text{ground,up}} + I_{2\text{,down}} = I_{1\text{,up}} + I_{1\text{,down}}$
- $I_{\text{ground,up}} = I_{1\text{,up}} + I_{1\text{,down}} - I_{2\text{,down}}$
- $I_{1\text{,down}} = I_{1\text{,up}}$ and $I_{2\text{,down}} = I_{2\text{,up}}$
- so $I_{\text{ground,up}} = 2 * I_{1\text{,up}} - I_{2\text{,up}}$

```
I_ground_up = 2 * I_1_up - I_2_up
T_ground = (I_ground_up / (epsilon * sigma))^0.25
```

Again, you could let R do more of the work for you by writing

```
I_1_down = I_1_up
I_ground_up = I_1_down + I_1_up - I_2_down
T_ground = (I_ground_up / (epsilon * sigma))^0.25
```

Answer: T_{ground} = 334.4 K.

We can also use algebra to observe that

$$\begin{aligned} I_{\text{ground,up}} &= 2 * I_{1\text{,up}} - I_{2\text{,up}} \\ I_{2\text{,up}} &= 2 I_{2\text{,up}} \end{aligned}$$

so

$$\begin{aligned} I_{\text{ground,up}} &= 4 I_{2\text{,up}} - I_{2\text{,up}} \\ &= 3 I_{2\text{,up}} \\ T_{\text{ground}} &= \sqrt[4]{\frac{3 I_{2\text{,up}}}{\varepsilon\sigma}} \\ &= \sqrt[4]{3} T_{\text{bare rock}}. \end{aligned}$$

In a 1-layer model, the ground temperature was $2^{0.25}$ times the bare-rock temperature, and in a 2-layer model, the bround temperature is $3^{0.25}$ times the bare-rock temperature.

Exercise 3.3

```
make_nuclear_winter_diagram()
```

Nuclear Winter. Let us go back to the One-Layer Model but change it so that the atmospheric layer absorbs visible light rather than allowing it to pass through (See the figure above). This could happen if the upper atmosphere were filled with dust. For simplicity, assume that the albedo of the Earth remains the same, even though in the real world it might change with a dusty atmosphere.> What is the temperature of the ground in this case?

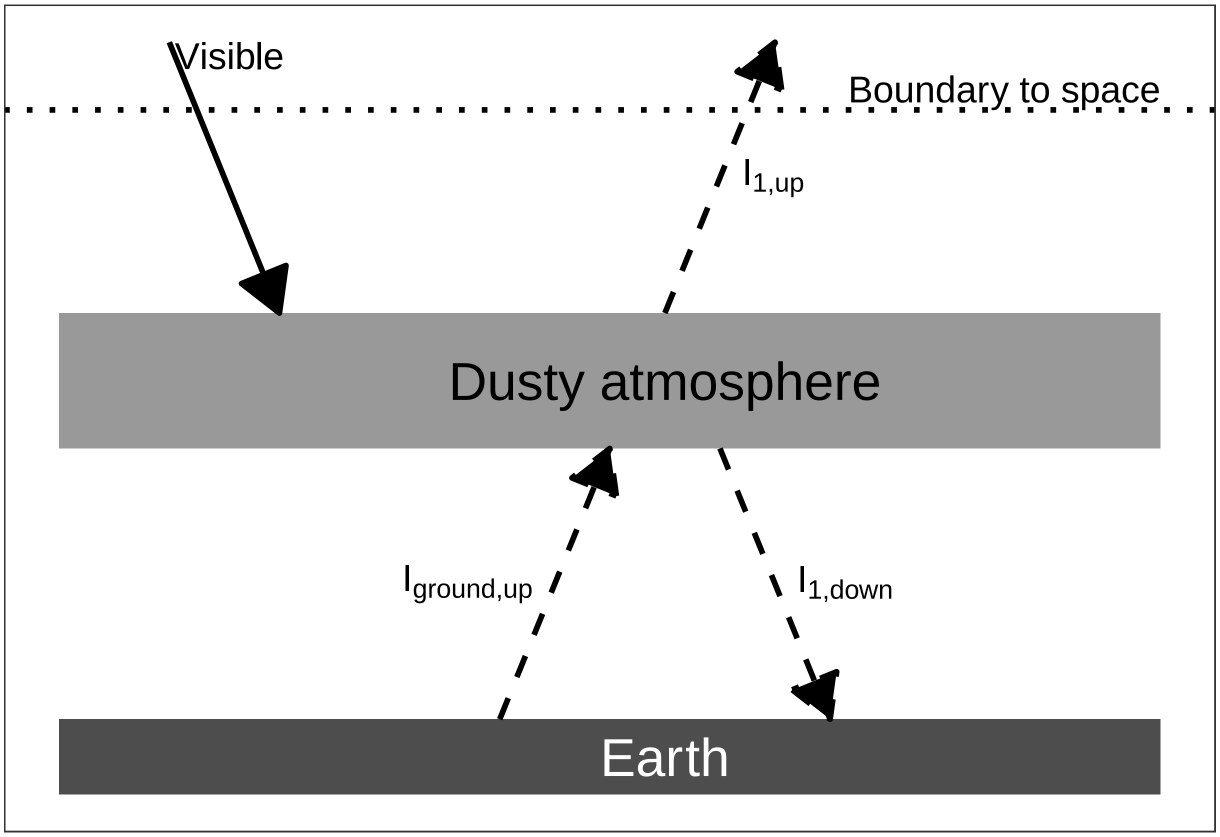


Figure 2: An energy diagram for a planet with an opaque pane of glass for an atmosphere. The intensity of absorbed visible light is $(1-\alpha)I_{solar}/4$.

Answer: Here, the key difference is that the heat from the sun is absorbed by the atmosphere instead of passing through the atmosphere to the ground.

The equation for the atmosphere is the same as in the 1-layer model because we use the energy balance at the boundary to space: $I_{\text{atm, up}} = I_{\text{visible}} = (1-\alpha)I_{\text{solar}}/4$ and the temperature of the atmosphere is the bare-rock temperature, just as the top layer of the atmosphere is for every layer model.

However, things are different at the ground. The energy balance at the dusty atmosphere is

- $\mathrm{I_{visible}}$ + $\mathrm{I_{ground,up}}$ = $\mathrm{I_{atm,up}}$ + $\mathrm{I_{atm,down}}$
- $\mathrm{I_{ground,up}}$ = $\mathrm{I_{atm,up}}$ + $\mathrm{I_{atm,down}}$ - $\mathrm{I_{visible}}$
- But $\mathrm{I_{atm,up}}$ = $\mathrm{I_{atm,down}}$ = $\mathrm{I_{visible}}$
- So $\mathrm{I_{ground,up}}$ = $\mathrm{I_{atm,up}}$.
- This means that $\mathrm{T_{ground}}$ = $\mathrm{T_{atmosphere}}$ = $\mathrm{T_{bare\text{-}rock}}$.

```
I_visible = (1 - alpha) * I_solar / 4
I_atm_up = I_visible
I_atm_down = I_visible
I_ground_up = I_atm_up + I_atm_down - I_visible
T_ground = (I_ground_up / (epsilon * sigma))^0.25
```

T_ground = 254.1 K. This is the same as the bare-rock temperature.

The effect of the dust in the atmosphere is to cancel out the greenhouse effect and cool off the surface to the bare-rock temperature. The greenhouse effect works because the atmosphere is transparent to shortwave light and opaque to longwave light. If the atmosphere becomes opaque to shortwave light, then the greenhouse effect doesn't work.